Developing Zynq Software with Xilinx SDK

Lab 11

Xilinx Libraries

August 2016
Version 08

Lab 11 Overview

Many standalone or baremetal applications take advantage of libraries of reusable code in order to reduce application development time and leverage existing, proven code.

Xilinx provides several libraries which can be built into a BSP as a selectable option. One of these libraries is the Xilinx In-system and Serial Flash (xilisf) Library which support the Xilinx In-system Flash and external serial Flash memories from Atmel, Winbond, Intel/ST Microelectornics/Micron/Numonyx, and Spansion devices. This library enables higher layer software (like an application) to communicate with the serial Flash device.

The library allows the user to Write, Read, and Erase the serial Flash. The user can also protect the data stored in the serial Flash from unwarranted modification by enabling device specific Sector Protection features.

The library supports interrupt driven mode and polled mode based upon the mode in which the lower level SPI driver is configured by the user. The library can also support multiple instance of serial Flash at a time provided that they are of the same device family as the device family selection made at compile time. A serial Flash operates as a slave device on the SPI bus with a Xilinx SPI core operating as the bus master. The library uses lower level SPI drivers to communicate with the serial Flash so using this library is an excellent way to abstract your application away from these lower level implementation details.

In this lab, an application which reads and writes MAC address (EUI-48 only) configuration data to/from the on-board QSPI Flash device is explored. For more information on what a MAC address is and why you might want to use one in an end application, please see the following Wikipedia article:

http://en.wikipedia.org/wiki/MAC_address

Lab 11 Objectives

When you have completed Lab 11, you will know:

- How to enable one of the Xilinx libraries within your standalone BSP
- How to leverage one of the Xilinx libraries from a standalone application

Experiment 1: Create the Application Project

Similar to the flow for importing existing application code in Lab 8, a new blank project is created and existing code is imported. The application code is then executed to verify the application code we were given functions as expected.

Experiment 1 General Instruction:

Create a new blank software application project. Import code from the following folder:

C:\Speedway\ZynqSW\2016_2\Support_documents\flash_mac_app\

Run the application on the target hardware and observe the behavior.

Experiment 1 Step-by-Step Instructions:

1. Launch Xilinx Software Development Kit (SDK) if not already open. **Start → All Programs → Xilinx Design Tools → Vivado 2016.2 → SDK → Xilinx SDK 2016.2**.

Figure 1 – The SDK Application Icon

2. Set or switch the workspace to the following folder and then click the **OK** button:

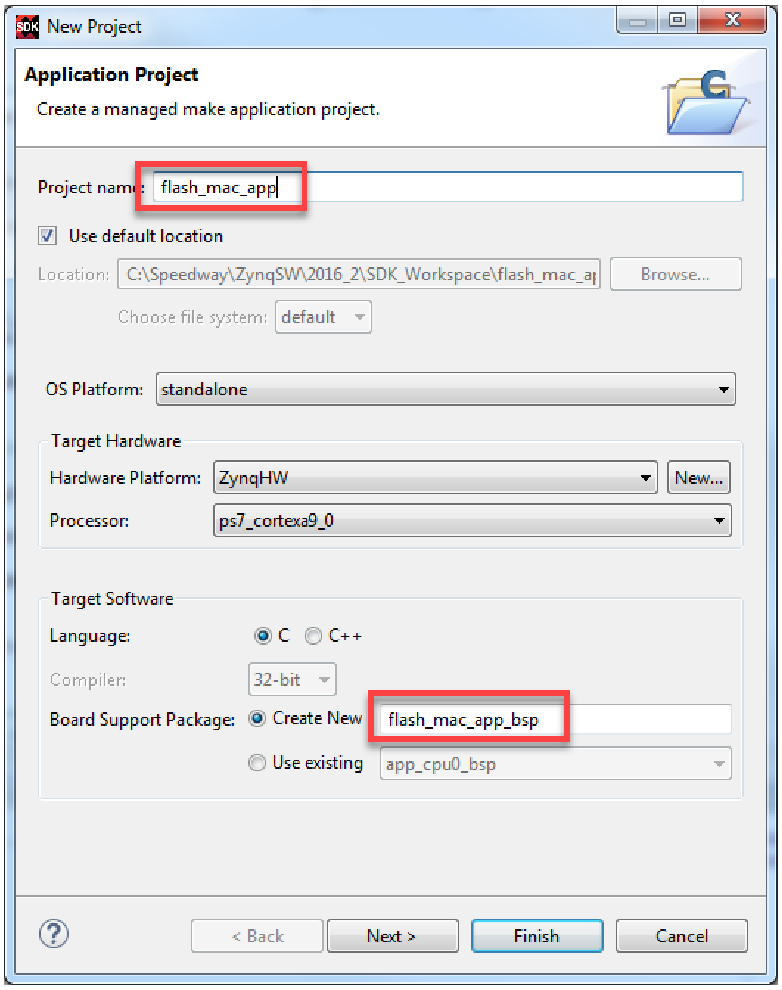
   C:\Speedway\ZynqSW\2016_2\SDK_Workspace\

3. Create a new SDK software application project by selecting the **File→New→ Application Project** menu item.

4. In the **New Project** wizard, change the **Project name** field to the **flash_mac_app** name.

   Change the **Board Support Package** to the **Create New** option and use the default **flash_mac_app_bsp** suggested name.

   Leave the other settings to their default values. Click the **Next** button to continue.



New Project

Application Project

Create a managed make application project.

Project name: flash_mac_app

[✓] Use default location

Location: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\flash_mac_a

Choose file system: default

OS Platform: standalone

Target Hardware

Hardware Platform: ZynqHW

Processor: ps7_cortexa9_0

Target Software

Language: [✓ C] C++

Compiler: 32-bit

Board Support Package: [✓ Create New] flash_mac_app_bsp

Use existing app_cpu0_bsp

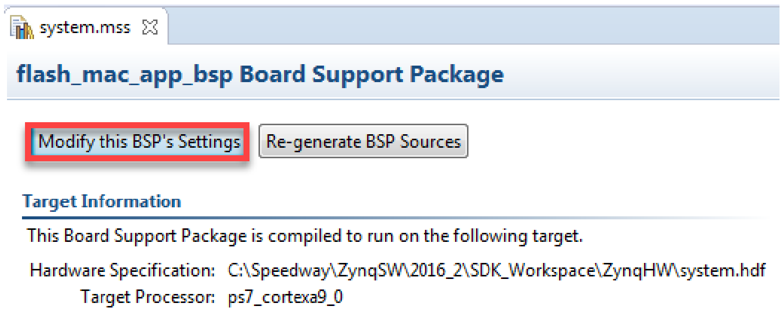
Figure 2 – Creating the flash_mac_app Application

5. Select the **Empty Application** project template and click the **Finish** button to complete the new project creation process using the Empty Application project template.

6. The empty **flash_mac_app** application project and **flash_mac_app_bsp** are created.

   Open the Board Support Package **system.mss** file if it is not already open in the Editor Pane.

7. Click on the **Modify this BSP's Settings** button to configure **flash_mac_app_bsp** settings.


system.mss

flash_mac_app_bsp Board Support Package

Target Information

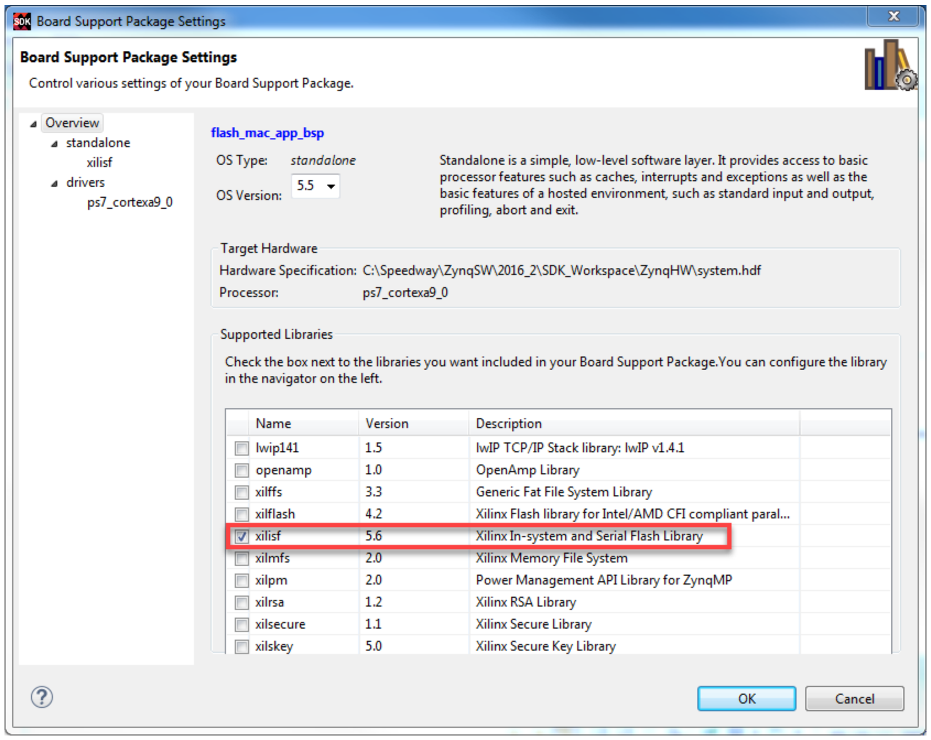
This Board Support Package is compiled to run on the following target.

Hardware Specification: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\ZynqHW\system.hdf

Target Processor: ps7_cortexa9_0

Figure 3 – Modify the BSP Settings for flash_mac_app_bsp Project

8. Enable the **Xilinx In-system and Serial Flash Library** for this BSP by selecting the checkbox next to the **xilisf** item in the **Supported Libraries** list.


Board Support Package Settings

Board Support Package Settings

Control various settings of your Board Support Package.

flash_mac_app_bsp

OS Type: standalone

OS Version: 5.5

Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit.

Target Hardware

Hardware Specification: C:\Speedway\ZynqSW\2016_2\SDK_Workspace\ZynqHW\system.hdf

Processor: ps7_cortexa9_0

Supported Libraries

Check the box next to the libraries you want included in your Board Support Package.You can configure the library in the navigator on the left.

	Name	Version	Description
☐	lwip141	1.5	lwIP TCP/IP Stack library: lwIP v1.4.1
☐	openamp	1.0	OpenAmp Library
☐	xilffs	3.3	Generic Fat File System Library
☐	xilflash	4.2	Xilinx Flash library for Intel/AMD CFI compliant paral...
✓	xilisf	5.6	Xilinx In-system and Serial Flash Library
☐	xilmfs	2.0	Xilinx Memory File System
☐	xilpm	2.0	Power Management API Library for ZynqMP
☐	xilrsa	1.2	Xilinx RSA Library
☐	xilsecure	1.1	Xilinx Secure Library
☐	xilskey	5.0	Xilinx Secure Key Library

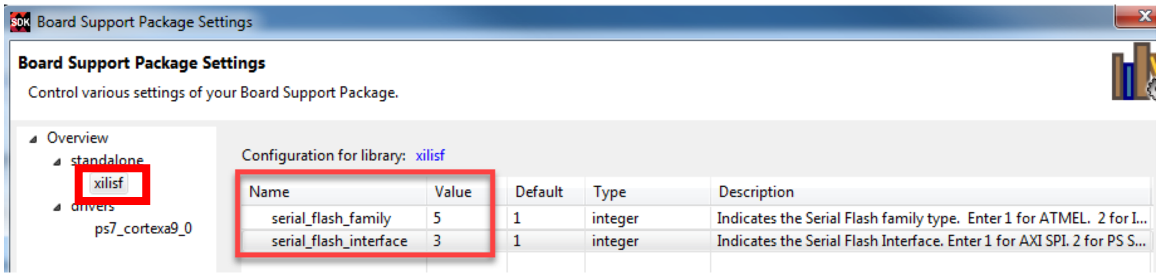
Figure 4 – Enable the Xilinx Library xilisf within the BSP Settings

9. Click on the **xilisf** listing under the tree to the left of the BSP Settings windows under **Overview→standalone**. This will open the **xilisf** library's specific settings.

   Since we are using a Spansion QSPI flash, according to the description for the **serial_flash_family** field, a value of **5** should be used to specify the Spansion Flash family type. Change the **Value** entry for this field to **5**.

Since we are using a Spansion QSPI flash, according to the description for the **serial_flash_interface** field, a value of **3** should be used to specify the QSPI Flash interface type. Change the **Value** entry for this field to **3**.

Click on the **OK** button to accept the updated BSP settings.


Board Support Package Settings
Board Support Package Settings
Control various settings of your Board Support Package.
Overview
standalone
xilisf
drivers
ps7_cortexa9_0

Configuration for library: xilisf

Name	Value	Default	Type	Description
serial_flash_family	5	1	integer	Indicates the Serial Flash family type. Enter 1 for ATMEL. 2 for I...
serial_flash_interface	3	1	integer	Indicates the Serial Flash Interface. Enter 1 for AXI SPI. 2 for PS S...

Figure 5 – Configure the Xilinx Library xilisf for Spansion QSPI Flash

10. In the **Project Explorer** tab, expand **flash_mac_app** and right-click on the **src** folder.

    Click on the **Import** option in the pop up menu.

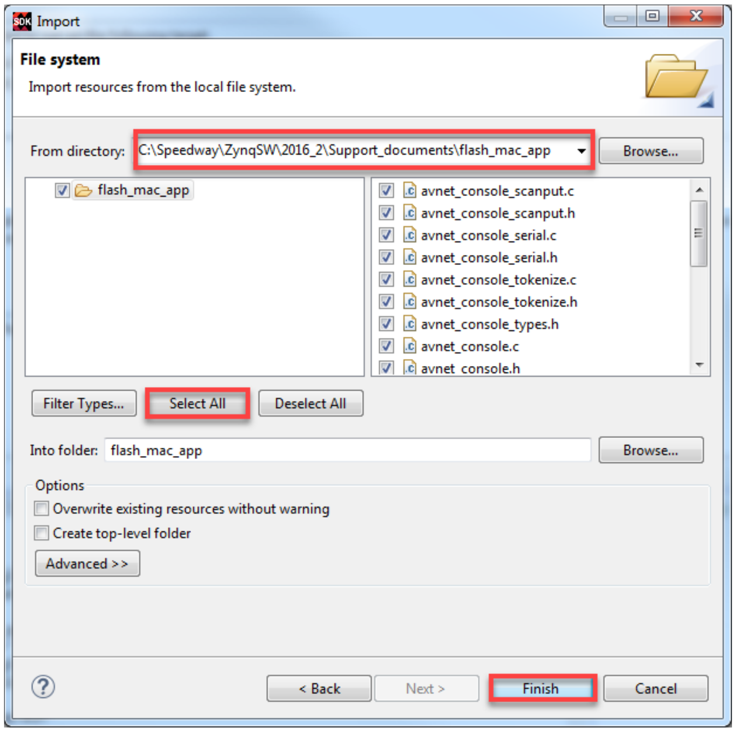
11. In the **Import** window, expand the **General** item, select the **File System** option, and click the **Next** button.

12. Click on the **Browse** button and select the following folder which contains the application code that we wish to start from:

    C:\Speedway\ZynqSW\2016_2\Support_documents\flash_mac_app\

    After this folder is selected within the **Browse** dialog, click the **OK** button to search the folder for files to import into the application project.

    Click the **Select All** button to select the 18 source code files and then click the **Finish** button to complete the **Import** operation.



From directory: C:\Speedway\ZynqSW\2016_2\Support_documents\flash_mac_app

[✓] flash_mac_app

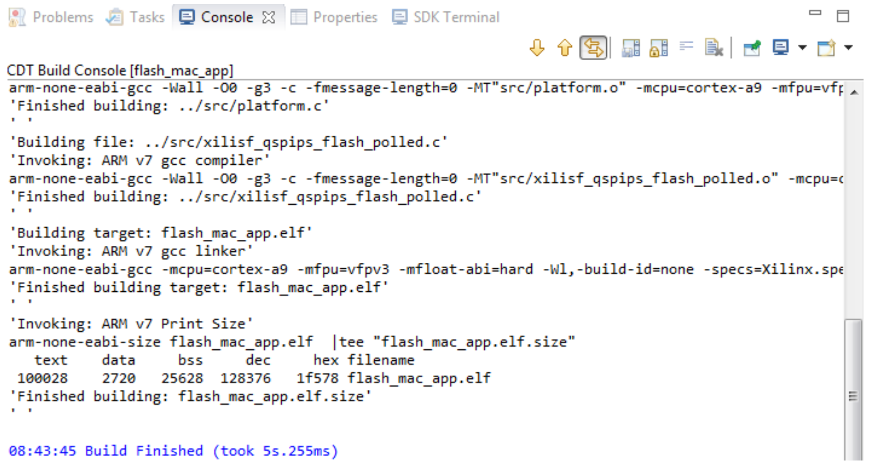
[✓] avnet_console_scanput.c
[✓] avnet_console_scanput.h
[✓] avnet_console_serial.c
[✓] avnet_console_serial.h
[✓] avnet_console_tokenize.c
[✓] avnet_console_tokenize.h
[✓] avnet_console_types.h
[✓] avnet_console.c
[✓] avnet_console.h

Into folder: flash_mac_app

Options
[ ] Overwrite existing resources without warning
[ ] Create top-level folder

Figure 6 – Selecting flash_mac_app Application Source Files

13. The SDK Console panel shows the results of the build. Make sure that the application is built without errors.



```
CDT Build Console [flash_mac_app]
arm-none-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0 -MT"src/platform.o" -mcpu=cortex-a9 -mfpu=vfp
'Finished building: ../src/platform.c'
' '
'Building file: ../src/xilisf_qspips_flash_polled.c'
'Invoking: ARM v7 gcc compiler'
arm-none-eabi-gcc -Wall -O0 -g3 -c -fmessage-length=0 -MT"src/xilisf_qspips_flash_polled.o" -mcpu=c
'Finished building: ../src/xilisf_qspips_flash_polled.c'
' '
'Building target: flash_mac_app.elf'
'Invoking: ARM v7 gcc linker'
arm-none-eabi-gcc -mcpu=cortex-a9 -mfpu=vfpv3 -mfloat-abi=hard -Wl,-build-id=none -specs=Xilinx.spe
'Finished building target: flash_mac_app.elf'
' '
'Invoking: ARM v7 Print Size'
arm-none-eabi-size flash_mac_app.elf  |tee "flash_mac_app.elf.size"
   text    data     bss     dec     hex filename
 100028    2720   25628  128376   1f578 flash_mac_app.elf
'Finished building: flash_mac_app.elf.size'
' '

08:43:45 Build Finished (took 5s.255ms)
```

Figure 7 – Application Build Console Window

14. After SDK finishes compiling the new application code, the ELF is available in the following location:

C:\Speedway\ZynqSW\2016_2\SDK_Workspace\flash_mac_app\Debug\flash_mac_app.elf

At this point the application is ready to be launched on the target hardware.

Questions:

Answer the following questions:

- *What if you wanted to reuse part of this application on your own Zynq design but you prefer to use Atmel Flash instead of Spansion Flash?*

____

- *Why is a separate BSP created for this application? Why not reuse the standalone_bsp_0 from earlier lab activities?*

____

Experiment 2: Exploring the Application

In this experiment, we will launch the application and experiment with reading and writing to the Spansion QSPI Flash device from the application.

This is accomplished by the use of the BSP library **xilisf** API. The following API calls are used to add serial flash support to an application:

```
int  Xisf_Initialize
     (XIsf *InstancePtr,
     XIsf_Iface *SpiInstPtr,
     u8 SlaveSelect,
     u8 WritePtr)
```

- The geometry of the underlying serial Flash device is determined by reading the Joint Electron Device Engineering Council (JEDEC) Device Information and the Status Register of the Serial Flash device.
- A blocking call which reads the JEDEC information of the Flash device and waits until the transfer is complete before checking to see if the information is valid.

```
int  Xisf_Erase
     (XIsf *InstancePtr,
     XIsf_EraseOperation Operation,
     u32 Address)
```

- Erases the contents of the specified memory in the serial Flash.

```
int  Xisf_Read
     (XIsf *InstancePtr,
     XIsf_ReadOperation Operation,
     void *OpParamPtr)
```

- Reads data from the serial Flash.

```
int  Xisf_Write
     (XIsf *InstancePtr,
     XIsf_WriteOperation Operation,
     void *OpParamPtr)
```

- Writes the specified data to the serial Flash.

Experiment 2 General Instruction:

Launch the **flash_mac_**app on the target hardware and experiment with reading the MAC address setting and storing a new MAC address.

Explore the application source code.

Experiment 2 Step-by-Step Instructions:

1. After completing Experiment 1, the application is ready to be launched on the target hardware.

2. *<ZedBoard & PicoZed Only>* Connect the power cable to the ZedBoard or PicoZed, but leave it powered OFF for now.

   <MicroZed Only> MicroZed must be powered OFF so that the new boot mode settings can take effect.

3. Connect the JTAG as follows:

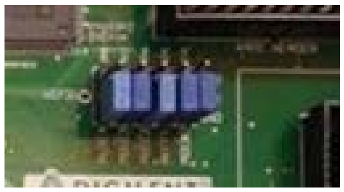
   a. ZedBoard – Connect a 2nd micro-USB cable between the host machine and connector J17 (JTAG)

   b. MicroZed -- Connect a Platform Cable or Digilent Programming cable from the host machine to the 2x7 JTAG socket on MicroZed, J3.

   c. PicoZed – Connect a Platform Cable or Digilent Programming cable from the host machine to the 2x7 JTAG socket on PicoZed, J7.

4. Set the Boot Mode jumpers to Cascaded JTAG Mode

   a. ZedBoard and MicroZed



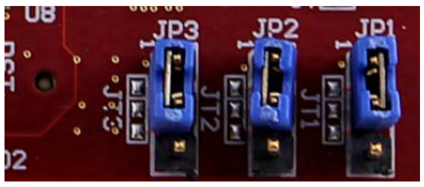


Figure 8 –Cascaded JTAG Mode : ZedBoard left; MicroZed right

   b. PicoZed

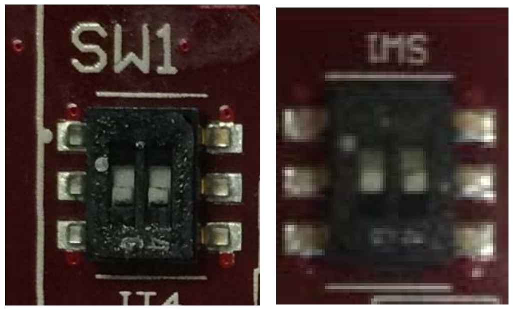


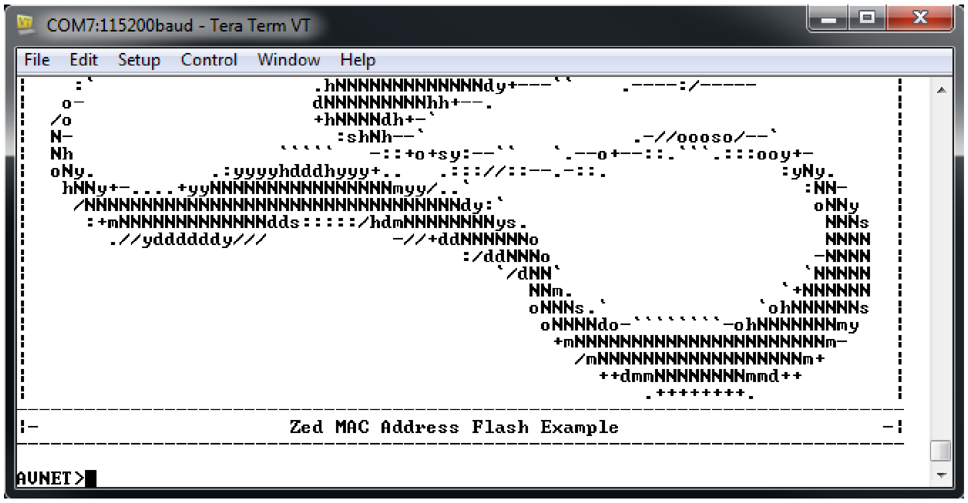
Figure 9 – PicoZed SW1 Set to JTAG Boot 7010/20 on the Left; 7015/30 on the Right

5. *<ZedBoard & PicoZed Only>* Turn power on.
6. Connect a micro-USB cable between the Windows Host machine and the USB-UART:
    a. ZedBoard connector J14 (UART)
    b. MicroZed connector J2. Since MicroZed gets power from the USB-UART, you should see the Green Power Good LED (D5) and the Red User LED (D3) light.
    c. PicoZed connector J1(USB_UART).
7. In SDK, select **Xilinx Tools → Program FPGA** or click the icon.
8. SDK will already know the correct .bit file (and .bmm if your future hardware platform includes that) since this was imported with the hardware platform. Click the **Program** button.

    When MicroZed D2 / PicoZed D3 or ZedBoard LD12 LED lights blue, the PL has configured successfully. Look for the message "`FPGA configured successfully with bitstream`" in the **SDK Log** window.
9. Launch a terminal program (Tera Term) with the 115200/8/n/1/n settings.
10. In the **Project Explorer** tab, right-click on the **flash_mac_app** project folder.

    Click on the **Run As→ Launch on Hardware (System Debugger)** option in the pop up menu.

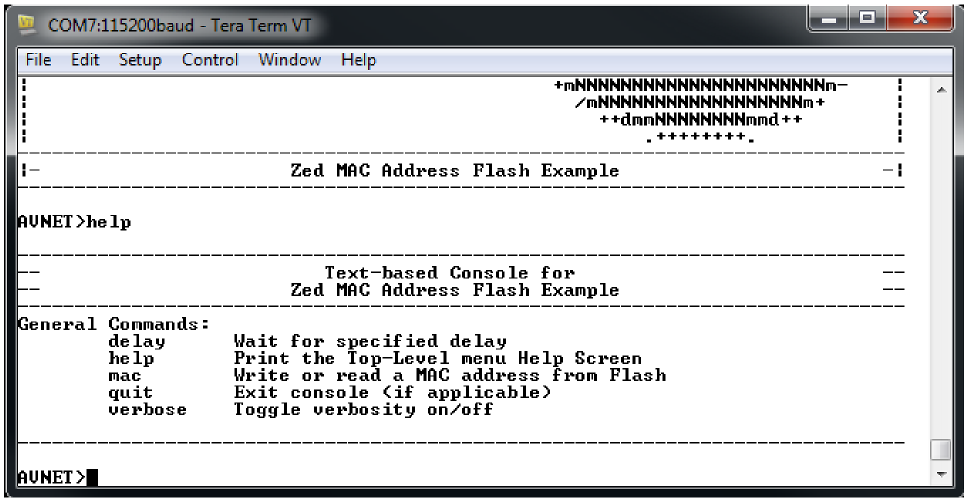
11. Using the Launch on Hardware option will automatically create a run configuration using default settings and launch the application on the target platform. Once the application is launched and is running on the target hardware, a command prompt should appear in the terminal window.


COM7:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
|  :`                        .hNNNNNNNNNNNNNNdy+---``     .----:/-----          |
|  o-                       dNNNNNNNNNNhh+--.                                    |
| /o                        +hNNNNdh+-`                                          |
| N-                          :shNh--`                  .-//oooso/--`            |
| Nh                    `````   -::+o+sy:--``   `.--o+--::.```.:::ooy+-          |
| oNy.         .:yyyyhdddhyyy+..    .:::/:-.-::.                :yNy.           |
|  hNNy+-....+yyNNNNNNNNNNNNNNNNmyy/..`                           :NN-           |
|   /NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNdy:`                        oNNy          |
|    :+mNNNNNNNNNNNNNNdds:::::/hdmNNNNNNNNys.                       NNNs          |
|      .//yddddddy///          -//+ddNNNNNNo                        NNNN          |
|                                    :/ddNNNo                      -NNNN          |
|                                       `/dNN`                     `NNNNN         |
|                                         NNm.                   `+NNNNNN         |
|                                         oNNNs.`              `ohNNNNNNs         |
|                                          oNNNNdo-````````-ohNNNNNNNmy           |
|                                           +mNNNNNNNNNNNNNNNNNNNNNNNNm-          |
|                                             /mNNNNNNNNNNNNNNNNNNNm+             |
|                                               ++dmmNNNNNNNNmmd++                |
|                                                   .+++++++++.                   |
--------------------------------------------------------------------------------
|-                        Zed MAC Address Flash Example                        -|
--------------------------------------------------------------------------------

AVNET>
```

Figure 10 – Initial Terminal Output from flash_mac_app Application

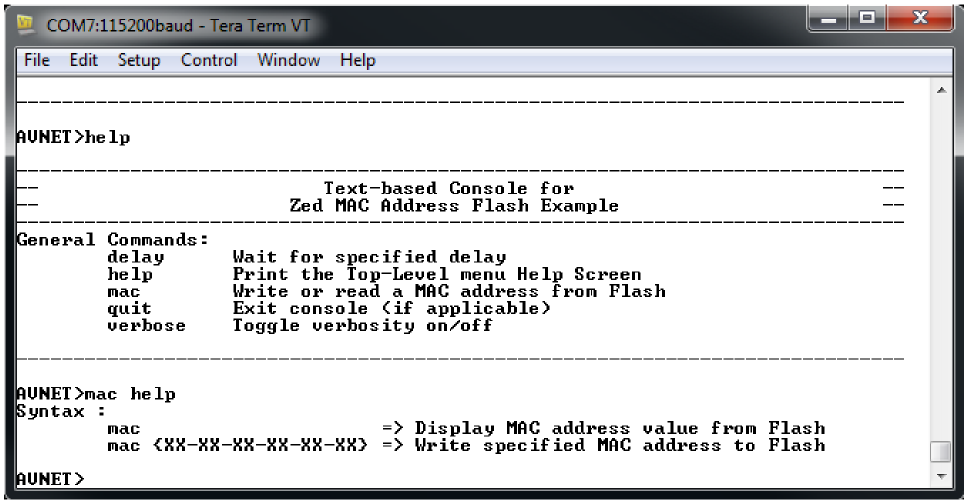
12. Type the command **help** at the prompt and press enter. This will display a listing of the commands supported by this application.


COM7:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
|                                           +mNNNNNNNNNNNNNNNNNNNNNNNNm-          |
|                                             /mNNNNNNNNNNNNNNNNNNNm+             |
|                                               ++dmmNNNNNNNNmmd++                |
|                                                   .+++++++++.                   |
--------------------------------------------------------------------------------
|-                        Zed MAC Address Flash Example                        -|
--------------------------------------------------------------------------------

AVNET>help

--------------------------------------------------------------------------------
--                           Text-based Console for                           --
--                        Zed MAC Address Flash Example                       --
--------------------------------------------------------------------------------
General Commands:
        delay      Wait for specified delay
        help       Print the Top-Level menu Help Screen
        mac        Write or read a MAC address from Flash
        quit       Exit console (if applicable)
        verbose    Toggle verbosity on/off

--------------------------------------------------------------------------------
AVNET>
```

Figure 91 – Initial Terminal Output from flash_mac_app Application

13. One of the commands listed is the **mac** command. Discover how to use this command by typing **mac help** at the prompt and then press enter. This will display the usage for the **mac** command.

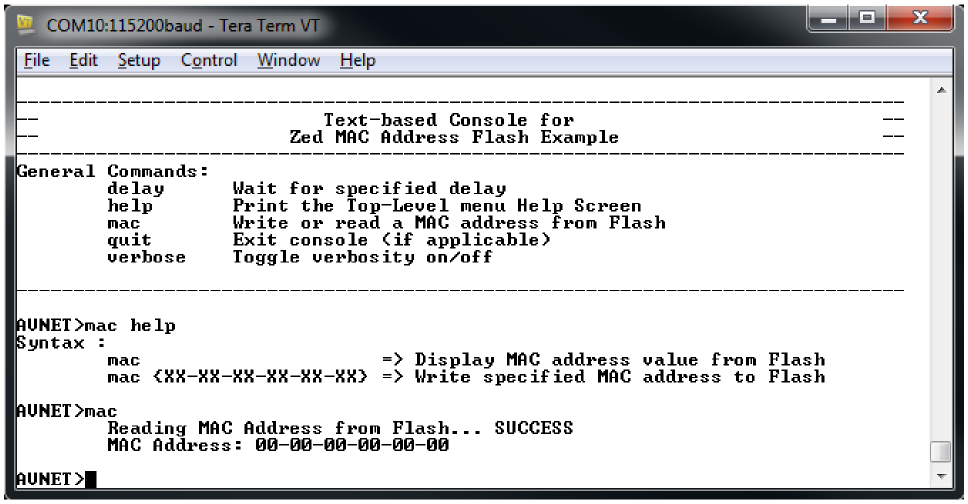

COM7:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
--------------------------------------------------------------------------------

AVNET>help

--------------------------------------------------------------------------------
--                          Text-based Console for                            --
--                        Zed MAC Address Flash Example                       --
--------------------------------------------------------------------------------
General Commands:
        delay      Wait for specified delay
        help       Print the Top-Level menu Help Screen
        mac        Write or read a MAC address from Flash
        quit       Exit console (if applicable)
        verbose    Toggle verbosity on/off

--------------------------------------------------------------------------------

AVNET>mac help
Syntax :
        mac                        => Display MAC address value from Flash
        mac <XX-XX-XX-XX-XX-XX>    => Write specified MAC address to Flash

AVNET>
```

Figure 102 – Initial Terminal Output from flash_mac_app Application

14. Use the **mac** command to read and display the current MAC address stored in the QSPI Flash.

    Since this is likely the first time this command has been run on memory, it is likely that a bunch of random characters are present in the board configuration data space we are using at **0x00090000**. In the example shown here, all zero data is found.

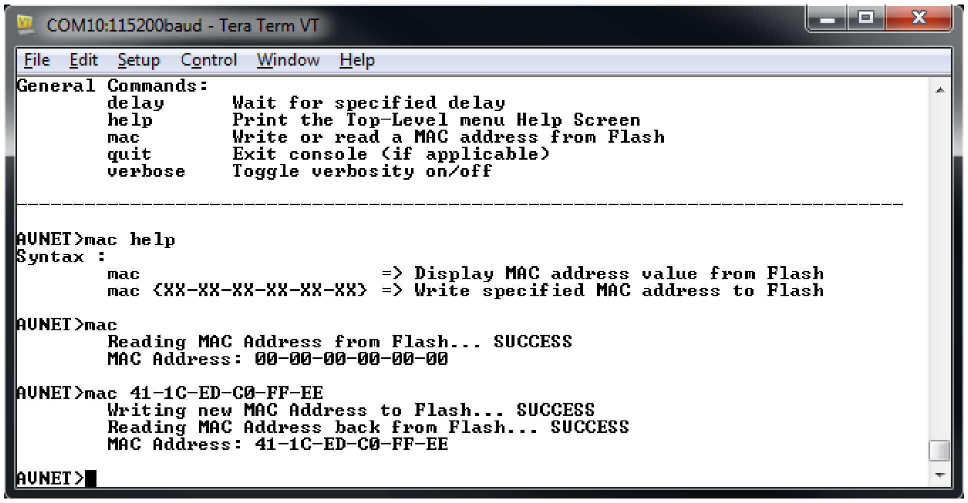

COM10:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
--------------------------------------------------------------------------------
--                          Text-based Console for                            --
--                        Zed MAC Address Flash Example                       --
--------------------------------------------------------------------------------
General Commands:
        delay      Wait for specified delay
        help       Print the Top-Level menu Help Screen
        mac        Write or read a MAC address from Flash
        quit       Exit console (if applicable)
        verbose    Toggle verbosity on/off

--------------------------------------------------------------------------------

AVNET>mac help
Syntax :
        mac                        => Display MAC address value from Flash
        mac <XX-XX-XX-XX-XX-XX>    => Write specified MAC address to Flash

AVNET>mac
        Reading MAC Address from Flash... SUCCESS
        MAC Address: 00-00-00-00-00-00

AVNET>
```

Figure 113 – Displaying Initial Flash Data in the MAC Address Space

15. Let's set a new MAC address for this board by again using the mac command. This time, the desired MAC address value will be specified using the following command:

```
AVNET> mac 41-1C-ED-C0-FF-EE
```


COM10:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
General Commands:
        delay      Wait for specified delay
        help       Print the Top-Level menu Help Screen
        mac        Write or read a MAC address from Flash
        quit       Exit console (if applicable)
        verbose    Toggle verbosity on/off

------------------------------------------------------------------------------

AVNET>mac help
Syntax :
        mac                         => Display MAC address value from Flash
        mac <XX-XX-XX-XX-XX-XX>     => Write specified MAC address to Flash

AVNET>mac
        Reading MAC Address from Flash... SUCCESS
        MAC Address: 00-00-00-00-00-00

AVNET>mac 41-1C-ED-C0-FF-EE
        Writing new MAC Address to Flash... SUCCESS
        Reading MAC Address back from Flash... SUCCESS
        MAC Address: 41-1C-ED-C0-FF-EE

AVNET>
```

Figure 124 – Programming a New MAC Address Value into Flash Memory

16. The new MAC address value is now programmed into the QSPI Flash. You can display the new value read back from the Flash with the **mac** command.

    If you are suspicious about the data being cached in volatile memory, you can also power cycle your board and re-run the application to demonstrate that the MAC address value is indeed stored in non-volatile Flash memory.

17. Let's explore how this application takes advantage of the XilISF library API to access the QSPI flash.

    Expand the **flash_mac_app → src** folder in the **Project Explorer** panel. Double click the application source file **flash_mac.c** to open it in the code editor.

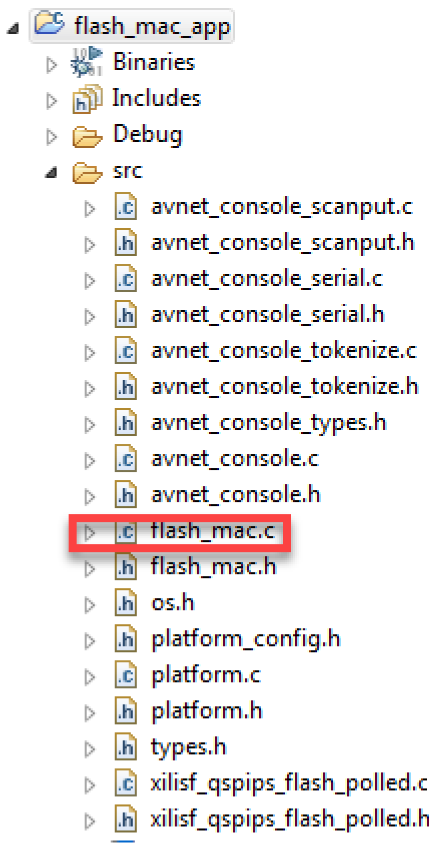


Figure 135 – flash_mac_app Source Code

18. In the source file, line 122 shows a call to **`FlashInit()`** which in turn makes a call to **`QsipiFlashPolledInit()`** which is code leveraged from this Xilinx example file:

    C:\Xilinx\SDK\2016.2\data\embeddedsw\lib\sw_services\xilisf_v5_0\examples\xilisf_qspips_stm_polled_example.c

    The code from the example application was adapted for the MAC address application we used in this lab with very little modification to achieve physical Flash data access.

19. There are a few additional routines which were written to support this application.

    The **`FlashErase()`** function allows data sectors to be erased in preparation for writing new data to the Flash memory. This function takes advantage of the **xilisf** library **`Xisf_Erase()`** API call.

The `FlashRead()` function provides support for reading MAC address data from the QSPI Flash. This function takes advantage of the **xilisf** library `Xisf_Read()` API call.

The `FlashWrite()` function provides support for writing MAC address data to the QSPI Flash. This function takes advantage of the **xilisf** library `Xisf_Write()` API call.

Questions:

Answer the following questions:

- *How were the functions within the application code which access the flash written? Was it written entirely from scratch?*

____

Exploring Further

If you have more time and would like to investigate more…

- Add another Xilinx library to your standalone BSP.
- Explore the examples provided for various software libraries within the respective example subfolders under the following directory:

C:\Xilinx\SDK\2016.2\data\embeddedsw\lib\sw_services\

This concludes Lab 11.

Revision History

Date	Version	Revision
12 Nov 13	01	Initial Release
25 Nov 13	02	Revisions after pilot
14 Dec 14	03	Review for update to Vivado 2014.3
07 Jan 15	04	Update to 2014.4

09 Mar 15	05	Merge MicroZed and ZedBoard instructions
18 Mar 15	06	Finalize SDK 2014.4
Oct 15	07	Updated to SDK 2015.2
Aug 16	08	Updated to SDK 2016.2

Resources

www.microzed.org

www.picozed.org

www.zedboard.org

www.em.avnet.com/drc

www.xilinx.com/zynq

www.xilinx.com/sdk

www.xilinx.com/vivado

www.xilinx.com/products/silicon-devices/soc/zynq-7000/ecosystem/index.htm

Answers

Experiment 1

- *What if you wanted to reuse part of this application on your own Zynq design but you prefer to use Atmel Flash instead of Spansion Flash?*

Avnet also distributes Atmel devices and the only part of this software application that would need to be changed is the **serial_flash_family** field from Experiment 1, Step 10.

- *Why is a separate BSP created for this application? Why not reuse the standalone_bsp_0 from earlier lab activities?*

A separate BSP was created since the Xilinx **xilisf** library was to be added specifically to support this application. We could have instead modified the **standalone_bsp_0** to add the **xilisf** library but that would cause the library code to become linked into our other standalone applications which rely upon **standalone_bsp_0** project as the BSP. This would unnecessarily increase the code size of those other standalone applications which might not be desirable.

Experiment 2

- *How were the functions within the application code which access the flash written? Was it written entirely from scratch?*

Not really, the flash access functions found within the **xilisf_qspips_flash_polled.c** source file are adapted from the example code found within the Xilinx **xilisf** library **C:\Xilinx\SDK\2016.2\data\embeddedsw\lib\sw_services\xilisf_v5_0\examples\** folder.